HW1 Dry

Dagger

Dagger injects dependencies in compile time by generating code, as opposed to Guice, which uses runtime reflection. Dagger API is much narrower than Guice's.
Configuring DI with Dagger:
First, module class is defined with @Module annotation. Instead of binding methods, configuration is done by annotating "binding" methods with @Provide, which can be a boilerplate from time to time. @Inject is used the same as in Guice (assuming you always annotate injectable classes with @Inject, because now you have to).

Module Structure

There is an hierarchical module structure. The root, AppComponent module, is composed of 6 modules, e.g one of them is ActivityBuilder, which binds a few bunches of modules together, each for different "sub-project". ChatRoomsFragmentProvider for example is bundled in the main activity, and it provides a ChatRoomsFragmentModule that defines the dependency leaves, one of them is our RocketChatClient.

RocketChatClient

Is created using RocketChatClientFactory singelton (provided in ChatRoomsFragmentModule as said above), and the factory itself depends on others. Different implementation can be injected by either editing the factory get method (or by providing a different factory, depends on what exactly we want to change).

CreateChannelView

createChannelView provider in CreateChannelModule should be changed to return the new implementation.